Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий

Кафедра «Вычислительная техника»

УТВЕРЖДАЮ
Заведующий кафедрой
_____ О.В. Непомнящий
подпись инициалы, фамилия
« ___ » _____ 20 ___ г

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

Разработка протокола маршрутизации на основе алгоритма роевого интеллекта
тема

09.04.01 Информатика и вычислительная техника
код и наименование направления

09.04.01.05 Сети ЭВМ и телекоммуникации
код и наименование магистерской программы

Научный руководитель	_____ подпись, дата	доцент, канд. техн. наук должность, ученая степень	Ф.А. Казаков инициалы, фамилия
Выпускник	_____ подпись, дата		К.Р. Васерчук инициалы, фамилия
Рецензент	_____ подпись, дата	профессор, канд. техн. наук должность, ученая степень	Ю.А. Маглинец инициалы, фамилия
Нормоконтролер	_____ подпись, дата		В.И. Иванов инициалы, фамилия

Красноярск 2018

Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий

Кафедра «Вычислительная техника»

УТВЕРЖДАЮ
Заведующий кафедрой
_____ О.В. Непомнящий
подпись инициалы, фамилия
« ___ » _____ 20 ___ г

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
в форме магистерской диссертации

Студенту Васерчук Кириллу Романовичу
Группа КИ16-01-5М Направление (специальность) 09.04.01 Информатика и вычислительная техника
Тема выпускной квалификационной работы Разработка протокола маршрутизации на основе алгоритма роевого интеллекта
Утверждена приказом по университету № 4280/с от 03.04.2017 г.
Руководитель ВКР Казаков Фёдор Александрович
Исходные данные для ВКР _____
_____
_____
Перечень разделов ВКР_____
_____
_____
_____
Перечень графического материала _____
_____
_____
_____

Руководитель ВКР _____ Ф.А. Казаков
подпись

Задание принял к исполнению _____ К.Р. Васерчук
подпись

« ___ » _____ 20__ г.

РЕФЕРАТ

Магистерская диссертация по теме «Разработка протокола маршрутизации на основе алгоритма роевого интеллекта» содержит 52 страницы текстового документа, 17 использованных источников, 8 иллюстраций, 7 таблиц, 2 формулы.

РОЕВОЙ ИНТЕЛЛЕКТ, ПРОТОКОЛ, СЕТЬ, МАРШРУТИЗАТОР, РОУТЕР, УЗЕЛ, ПУТЬ, КАНАЛ, ИНТЕРФЕЙС, МЕТРИКА, ПРОПУСКАНАЯ СПОСОБНОСТЬ, ТАБЛИЦА МАРШРУТИЗАЦИИ.

Объект, подлежащий разработке – сетевой протокол роевого интеллекта.

Цель работы – разработка протокола маршрутизации на основе роевого интеллекта, учитывающего загрузку каналов связи, и на основании этого параметра, строящего маршруты для передачи данных между узлами сети.

Задачи:

− провести сравнительный анализ алгоритмов роевого интеллекта;

− разработать граф работы протокола роевого интеллекта;

− разработать протокол маршрутизации, алгоритм работы которого будет основан на алгоритме роевого интеллекта;

− провести моделирование разработанного протокола в среде сетевого моделирования OMNeT++.

В результате рассмотрения существующих алгоритмов роевого интеллекта был разработан алгоритм работы протокола роевого интеллекта AntNet. Были сформированы граф работы протокола, форматы служебных пакетов. В ходе моделирования был получен график пропускной способности роутера, использующего протокол AntNet.

В итоге был разработан и промоделирован протокол маршрутизации, алгоритмы работы которого основаны на алгоритме роевого интеллекта. В основу алгоритма работы нового протокола лег «Муравьиный алгоритм».

СОДЕРЖАНИЕ

ВВЕДЕНИЕ 3
1 Роевой интеллект 6
2 Ant алгоритм 12
2.1 Описание 12
2.1.1 Решение при помощи алгоритма роевого интеллекта 12
2.1.2 Идея мобильных агентов 17
2.2 Протокол 18
2.3 Граф работы протокола 21
3 Реализация AntNet 23
3.1 Обзор среды OMNeT++ 23
3.2 OMNeT++ модель 24
3.3 Алгоритм протокола 27
3.3.1 Алгоритм работы протокола AntNet 27
3.3.2 Инициализация сети 28
3.3.3 Поддержание оптимальных маршрутов 32
3.3.4 Обработка сбоев 36
3.4 Структуры пакетов 39
4 Моделирование 42
4.1 Результаты эксперимента 42
ЗАКЛЮЧЕНИЕ 46
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 48

ВВЕДЕНИЕ

Маршрутизация – это процесс определения на основе данных из таблицы маршрутизации оптимального пути от узла-источника к узлу-получателю в условиях избыточных связей [2].

В процессе построения маршрутизации выделяют две части: определение дальнейшего пути пакета и непосредственно его пересылка по этому путь [3].

В соответствии с этими частями, процесс маршрутизации можно разделить на два иерархически связанных уровня:

− уровень маршрутизации. На этом уровне происходит работа с таблицей маршрутизации. Таблица маршрутизации служит для определения сетевого адреса следующего маршрутизатора, на который будет передан пакет, или непосредственно получателя пакета. После определения адреса передачи выбирается определенный выходной физический порт маршрутизатора для передачи сетевого пакета. Этот процесс называется определением маршрута перемещения пакета. Настройка таблицы маршрутизации ведется протоколами маршрутизации. На этом же уровне определяется перечень необходимых предоставляемых сервисов [3];

− уровень передачи пакетов. Перед тем как передать пакет, необходимо проверить контрольную сумму заголовка пакета, определить адрес (канального уровня) получателя пакета и произвести непосредственно отправку пакета с учетом очередности, фрагментации, фильтрации и прочих действий. Эти действия выполняются на основании команд, поступающих с уровня маршрутизации [3].

Маршрутизация является одной из наиболее важных процедур передачи данных. Процедура маршрутизации гарантирует, что данные перемещаются из одной сети в другую с оптимальной скоростью и минимальной задержкой. Целостность передаваемых данных в процессе построения маршрутизации гарантирована [4].

Алгоритм маршрутизации − часть программного обеспечения сетевого уровня, и отвечает за определение по какой линии отправлять пакет дальше. В независимости от того выбирается ли маршрут для сессии или для каждого пакета алгоритм маршрутизации должен обладать рядом свойств: корректностью, простотой, устойчивостью, стабильностью, справедливостью и оптимальностью [11].

Не адаптивные алгоритмы не принимают в расчет текущую загрузку сети и состояние топологии. Все возможные маршруты вычисляются заранее и загружаются в маршрутизаторы при загрузке сети. Такая маршрутизация называется статической маршрутизацией.

Адаптивные алгоритмы маршрутизации, наоборот, определяют маршрут исходя из текущей загрузки сети и топологии. Адаптивные алгоритмы различаются тем, где и как они получают информацию (локально от соседних маршрутизаторов или глобально ото всех), когда они меняют маршрут (каждые ΔT секунд, когда меняется нагрузка, когда меняется топология), какая метрика используется при оптимизации (расстояние, число скачков, ожидаемое время передачи) [11].

Маршрутизация по вектору расстояния − это алгоритм маршрутизации, идея которого в том, что у каждого маршрутизатора в подсети есть таблица расстояний до каждого маршрутизатора в подсети. Периодически маршрутизатор обменивается информацией со своими соседями и обновляет информацию в таблице. Каждый элемент таблицы состоит из двух полей: первое − номер линии, по которой надо отправлять пакеты, чтобы достичь нужного места, второе − величина задержки до места назначения. Эта величина задержки может быть измерена в разных единицах: скачках, миллисекундах, длине очереди на линии и т.д [11].

Алгоритм AntNet является адаптивным алгоритмом маршрутизации, метод поиска путей которого основан на поведении муравьев в природе. В нем предусмотрено исследование состояния сети с использованием агентов

(муравьев), которые в классическом представлении данного алгоритма, используют вероятностные правила выбора маршрутов. Для маршрутизации пакетов данных используются вероятностные таблицы маршрутизации [13].

Процедура поиска кратчайших путей в алгоритме AntNet использует те же принципы, что и колония муравьев при поиске пищи [12]. Муравей на своем пути оставляет след из активных веществ − феромонов. Феромоновый след во внешней среде существует некоторое время, и муравьи, которые будут идти следом, с больше вероятностью предпочтут то направление, по которому до них прошли другие муравьи. Эта вероятность будет тем больше, чем большее количество муравьев прошло по этому пути. Таким образом, через некоторое время большая часть муравьев будет использовать один, наиболее близкий к оптимальному, маршрут [12].

Роль муравьев в протоколе AntNet выполняют активные агенты. Активный агент в протоколе AntNet − это специальный пакет, который несет с собой статистику о состоянии пройденных сетевых каналов.

В произвольные моменты времени роутер сети, настроенной с помощью AntNet, генерирует сообщения, которые отсылаются до определенного принимающего узла, но каждое сообщение следует по своему маршруту. С помощью такого алгоритма замеряется время следования сообщения на каждом маршруте, и затем путь с меньшим временем следования сменяет прежний маршрут и становится новым маршрутом по умолчанию до данного узла назначения.

1 Роевой интеллект

Сам термин «Роевой интеллект» был введён учеными Ван Цзином и Херардо Бени в 1989 году. Понятие роевого интеллекта все теснее переплетается с алгоритмами обработки и оптимизации больших количеств и потоков информации. И для проверки актуальности роевого интеллекта, ранее, были поставлены следующие цели:

− рассмотреть основные алгоритмы роевого интеллекта;

− провести сравнительный анализ методов реализации РИ;

− проверить актуальность развития алгоритмов реализации.

Ещё давно люди стали интересоваться так называемым «роевым поведением» − каким образом птицы летят на юг огромными косяками, не сбиваясь с курса. Как огромные колонии муравьёв работают так слаженно и возводят структуры, по сложности не уступающие современным мегаполисам. Как пчёлы могут так точно определять и добывать в необходимом для всей колонии питание. Все эти большие группы животных/насекомых можно объединить одним общим словом − рой.

Но человечество не стоит на месте и, развиваясь, люди стали изобретать компьютеры, при помощи которых инженеры стали моделировать «роевой интеллект» (РИ) − попытки сделать роботизированные, автоматические и автоматизированные рои. Хоть далеко не все попытки были успешными, но таким образом, они положили начало созданию роевого интеллекта, заложив к его основанию некоторые фундаментальные правила. Одним из них является тот факт, что для роевого интеллекта необходимо большое (достаточно) количество агентов, способных взаимодействовать между собой и окружающей их средой локально. Наблюдая за различными естественными примерами роёв, человечество придумало различные модели роевого интеллекта, чьё поведение основывалось на различных путях взаимодействия с окружающей средой и между собой [10].

Модель роевого интеллекта подразумевает наличие так называемой «многоагентной системы», которая определяется как система, состоящая из множества интеллектуальных агентов − программ, способных самостоятельно на протяжении некоторого, достаточно длительного промежутка времени, выполнять поставленную задачу. На рисунке 1 изображены разнообразные методы реализации роевого интеллекта.

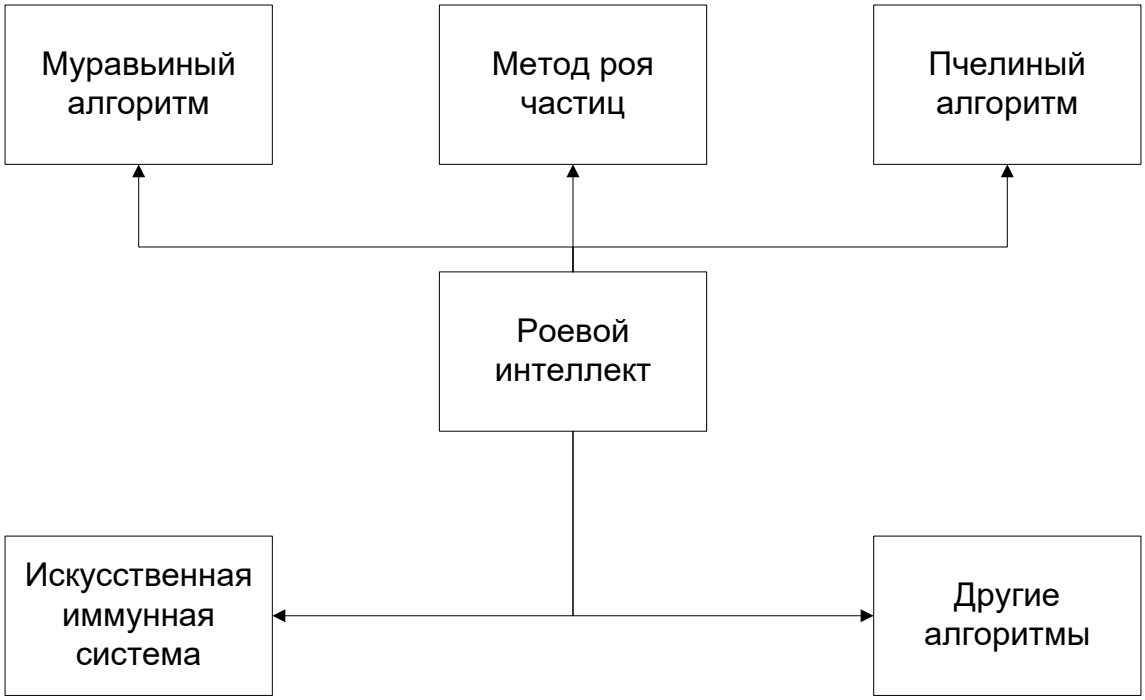

Рисунок 1 – Классификация структурных методов роевого интеллекта

Муравьиные алгоритмы представляют собой вероятностную жадную эвристику, где вероятности устанавливаются, исходя из информации о качестве решения, полученной из предыдущих решений. Они могут использоваться как для статических, так и для динамических комбинаторных оптимизационных задач. Сходимость гарантирована, то есть в любом случае будет получено оптимальное решение, но скорость сходимости алгоритма неизвестна [15].

Муравьиный алгоритм будет подробно рассмотрен далее. Рассмотрим другие алгоритмы.

«Метод роя частиц» является методом численной оптимизации, поддерживающий общее количество возможных решений, которые называются частицами или агентами, и перемещая их в пространстве к наилучшему найденному в этом пространстве решению, всё время находящемуся в изменении из-за нахождения агентами более выгодных решений [13].

Если проводить аналогию со стаей, то можно сказать, что все агенты алгоритма (частицы), в стае они могут быть птицами или рыбами, ставят для себя три довольно простых задачи:

– все агенты должны избегать пересечения с окружающими их агентам;

– каждая частица должна корректировать свою скорость в соответствии со скоростями окружающих её частиц;

– каждый агент должен стараться сохранять достаточно малое расстояние между собой и окружающими его агентами.

Алгоритм искусственной пчелиной колонии (или пчелиный алгоритм) – это алгоритм роевого интеллекта, основанный на имитации поведения колонии пчел. Алгоритм может использоваться в задачах оптимизации. Необходимым условием для его применения является наличие некоторого топологического расстояния или его аналога на области решений [16].

Работу алгоритма можно разбить на два этапа: инициализация и локальный поиск.

«Искусственная иммунная система» − это такая вычислительная система, которая способна адаптироваться и использовать схожие с реальной иммунной системой принципы и механики [17]. У данного метода есть три основные теории, которые описывают его функционирование и взаимодействие между элементами:

– теория отрицательного отбора;

– теория иммунной сети;

– теория клональной селекции.

Другие алгоритмы роевого интеллекта наименее всего распространены и редко используются. Перечень таких алгоритмов:

– метод капель воды, находящий либо наиболее близкие, либо наиболее оптимальные «пути для воды», подобно рекам;

– алгоритм кукушки − основан на паразитировании, подобно тому, как некоторые виды кукушек откладывали яйца в чужие гнёзда, со временем научившись имитировать цвета чужих яиц;

– метод альтруизма, основанный на том, что каждый агент «заботится» об окружающих, не обращая внимания на себя;

– метод гравитационного поиска − заключён в соблюдении закона всемирного тяготения (все тела притягиваются друг к другу), а именно в поисках наиболее качественных, «тяжёлых», агентов.

В ходе изучения основных алгоритмов роевого интеллекта («Муравьиный алгоритм», «Метод роя частиц», «Алгоритм пчелиной колонии» и «Алгоритм искусственной иммунной системы» (ИИС)) было проведено сравнение алгоритмов, выявлены их преимущества, сферы использования и полученные развития алгоритма. Полученные результаты иллюстрирует следующая таблица (Таблица 1) [14].

Таблица 1 – Сравнение алгоритмов роевого интеллекта

	Муравьиный алгоритм	Метод роя частиц	Пчелиный алгоритм	ИИС
Преимущества	• Достаточно эффективен для TSP (Traveling Salesman Problem) с небольшим количеством узлов; • Используется в приложениях, которые могут адаптироваться к изменениям; • Благодаря памяти всей колонии и случайному выбору пути не так сильно подвержен неудачным первоначальным решениям.	• Крайне низкая алгоритмическая сложность в реализации; • Достаточно эффективен для глобальной оптимизации.	• Возможность эффективного разделения на параллельные процессы; • Высокая скорость работы.	• Параметрическая пластичность; • Являются весьма гибкими по отношению к изменчивой внешней среде.
Применение	• Расчеты компьютерных и телекоммуникационных сетей; • Задача коммивояжёра; • Задача раскраски графа; • Задача оптимизации сетевых трафиков.	•Задачи машинного обучения; •Задачи оптимизации функций многих параметров, форм, размеров и топологий; •Биоинженерия, биомеханика, биохимия.	• Оптимизация управления; • Оптимизация классификаторов.	• Обеспечение компьютерной безопасности; • Распознавание образов; • Многомерная многоэкстремальная оптимизация; • Обнаружение аномалий во временных рядах данных.
Развитие	• Гибридизация с генетическими алгоритмами; • Использование базы нечётких правил.	• Представление МРЧ как многоагентной вычислительной системы; •Возможности включения других, более сложных методов РИ.	• Снижение зависимости от устанавливаемых параметров; • Объединение с генетическими алгоритмами.	• Клональный алгоритм отбора; • Негативный алгоритм отбора; • Иммунный сетевой алгоритм; • Дендритный алгоритм.

Рассмотрев все основные используемые алгоритмы роевого интеллекта: метод роя частиц, муравьиный алгоритм, алгоритм искусственной пчелиной колонии и алгоритм искусственной иммунной системы было принято решение использовать муравьиный алгоритм роевого интеллекта. Был проведен сравнительный анализ алгоритмов, в результате которого были выявлены основные сильные стороны методов, области их применения а также перспективы развития каждого, и наиболее подходящим оказался муравьиный алгоритм. Данный алгоритм и будет использоваться для разработки протокола маршрутизации, основанного на алгоритме роевого интеллекта.

2 Ant алгоритм

2.1 Описание

2.1.1 Решение при помощи алгоритма роевого интеллекта

Оптимизационный алгоритм с подражанием колонии муравьев (или муравьиный алгоритм) – один из наиболее эффективных алгоритмов для решения задач по поиску маршрутов в графах и по нахождению приблизительных решений для задачи коммивояжёра. Суть алгоритма заключается в применении модели функционирования колонии муравьев к решению различных задач. В этом алгоритме муравьиная колония рассматривается как мультиагентная система, в которой все агенты действуют самостоятельно по очень простым алгоритмам, но вся система в целом ведет себя крайне разумно. Поведение колонии муравьев основывается на самоорганизации, достигаемой за счет взаимодействия агентов на низком уровне ради общей цели. Особи могут взаимодействовать как с помощью прямого обмена информацией (химический, визуальный контакт), так и с помощью непрямого обмена (стигмержи). Он заключается в том, что некий агент может изменять область пространства с помощью некоторого вещества (феромона), после чего другие агенты могут использовать эту информацию для определения собственного маршрута. В результате концентрация феромонов на маршруте определяет приоритет его выбора. Кроме того, «феромон» может испаряться, что создает динамичность в данном алгоритме.

Первым, кто сумел применить поведение муравьев для решения задачи о поиске кратчайшего пути, стал Марко Дориго в начале 90-х годов XX века. Позже также были решены многие оптимизационные задачи при помощи муравьиных алгоритмов. В настоящее время эти алгоритмы показывают лучшие результаты в некоторых задачах. Концепция алгоритма заключается в

способности муравьев находить кратчайший путь крайне быстро и адаптироваться к различным внешним условиям. При движении каждый муравей помечает свой путь феромоном, что в дальнейшем используется другими муравьями. Это и есть простой алгоритм одного агента, который в сумме всех агентов колонии позволяет находить кратчайший путь или изменять его при обнаружении препятствия. Данную концепцию можно увидеть на рисунке 2.

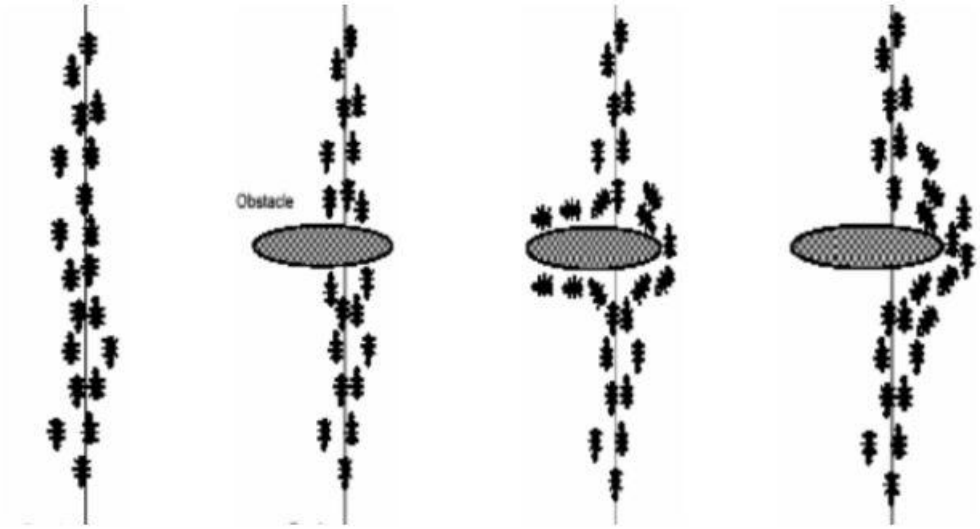

Рисунок 2 − Пример нахождения муравьями нового пути при появлении препятствия

Рассмотрим случай, когда на пути от муравейника к источнику пищи встречается развилка и муравей должен самостоятельно выбрать один из двух путей. Один из путей намного длиннее, но муравей, находящийся на развилке, не имеет об этом никакого представления. Через некоторое время оказывается, что так называемый «муравьиный трафик» направлен именно по короткому пути [3].

В определенном случае первый муравей, дойдя до развилки, выбирает неоптимальный, длинный путь. Муравей на пути своего следования помечает

пройденную дорогу, выделяя вещество под названием феромон. Феромоны помогают муравью найти обратный путь, выполняя, таким образом, роль зарубок, которые охотник оставляет на деревьях в лесу. Следующий муравей, дойдя до развилки дорог на пути домой, чувствует запах феромонов, оставленных первым муравьем. Чаще всего муравей выбирает путь, уже пройденный его сородичем. Но в муравьиной колонии есть особи, которые действуют противоположно большинству представителей колонии, и выбирают путь, на котором запах феромона является слабее или вообще отсутствует. Не исключено, что такой муравей решится пойти по неизведанному пути, который является непредпочтительным для большинства муравьев, и этот муравей повернет туда, где ранее еще никто не ходил. У следующего муравья выбор будет сложнее: следовать по длинной дороге, по которой проследовали уже многие, либо выбрать короткий путь, по которому прошел только один муравей и запах феромона является пока еще относительно слабым. Даже если новый муравей снова выберет традиционный маршрут – это не будет являться проблемой. Вероятность использования альтернативного пути все равно постепенно будет увеличиваться, так как феромоны постепенно испаряются, а муравьи, которые следует по короткому пути, успевают обновлять метки на коротком пути [5].

Каждый муравей, который выбирает короткий путь на развилке, постепенно повышает шансы, что по этому короткому пути пойдут и его последователи. Таким образом, происходит срабатывание механизма положительной обратной связи, что в итоге приводит к тому, что основной поток муравьев постепенно переключается с длинного пути на короткий. Традиции муравьиной колонии, таким образом, делают благое дело, сокращая тем самым расстояние от дома до источника пищи, а отдельные «исследователи», не признающие догм колонии, выбирающие дорогу, по

которой никто не ходит, либо добираются до источника пищи дольше остальных муравьев, либо вовсе остаются ни с чем [5].

Муравьиный алгоритм можно представить в виде следующего набора команд:

1. создание агентов;

2. поиск подходящего решения;

3. изменение феромона;

4. вспомогательные действия (не обязательно).

Рассмотрим каждый из шагов подробнее:

1. создание агентов:

– начальное расположение, где размещается агент, зависит от начальных условий и ограничений задачи. Агенты могут или быть в одной точке, или в разных с повторениями, или в разных без повторений;

– также указывается первоначальное значение феромона, чтобы значения не были нулевыми.

2. поиск подходящего решения:

– вероятность того, что произойдет переход из вершины i в j, можно определить по формуле [12]:

$$P_{ij}(t) = \frac{\tau_{ij}(t)^{\alpha}(\frac{1}{d_{ij}})^{\beta}}{\sum_{j\,\epsilon\,allowed\,nodes}\tau_{ij}(t)^{\alpha}(\frac{1}{d_{ij}})^{\beta}},$$

где $\tau_{ij}(t)$ − уровень феромона;

d_{ij}– эвристическое расстояние;

α, β − константные параметры.

– если α = 0, с большей вероятностью выберется ближайшая вершина;

– если β = 0, выбор будет основываться лишь на феромоне;

Необходим найденный экспериментальным способом компромисс между этими двумя величинами.

3. изменение феромона:

– уровень феромона изменяется по формуле [12]:

$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + \sum_{k\ \epsilon\ Colony\ that\ used\ edge\ (i,j)} \frac{Q}{L_k(t)},$$

где ρ – интенсивность испарения;

$L_k(t)$ – цена текущего решения k-го муравья;

Q – параметр, который имеет значение порядка цены оптимального решения;

$\frac{Q}{L_k(t)}$ – феромон, который откладывается k-м муравьем, который использует ребро.

4. вспомогательные действия: в основном используют алгоритмы локального поиска.

Эксперименты и наблюдения за природой живых организмов часто приводят к необычным и интересным техническим решениям. Так произошло и с такой областью исследований как «маршрутизация»: именно муравьиный алгоритм работы подсказал ученым Гианни Ди Каро (Gianni Di Caro) и Марко Дориго (Marco Dorigo) идею создания системы маршрутизации, основанную на базе мобильных агентов. Данная система была описана в 1998 году в совместной работе Ди Каро и Доринго.

Создание системы мобильных агентов было не первой попыткой использования муравьиного алгоритма к задачам маршрутизации. Шондерверд (Schoonderwoerd) и его коллеги в 1996-1997 годах начали рассматривать маршрутизацию как область исследований, для которой можно применить идею муравьиного алгоритма. Ими был придуман подход к управлению сетью при помощи муравьев (ABC − ant-based control). Этот подход используется в телефонных сетях при маршрутизации данных, и данный подход во многом отличается от алгоритма работы системы, придуманной Ди Каро и Доринго. Кроме того, ученые Субраманиан (Subramanian), Чен (Chen) и Друшель (Druschel) в 1997 году предложили использовать свой алгоритм, в основу

которого легло поведение муравьиной колонии в природе. Этот алгоритм работы был применен к сетям с коммутацией пакетов и представлял собой расширение ABC-алгоритма, идея которого принадлежала Шондерверду.

2.1.2 Идея мобильных агентов

Под программным агентом в компьютерных науках понимается программа, которая вступает в состояние посредничества с пользователем или другой целевой программой. Слово «агент» происходит от латинского слова agere (делать) и означает соглашение выполнять какие-то действия от имени кого-либо. Такие «действия от имени» означают право решать, какие действия (если они требуются) являются целесообразными для применения в системе. Основная идея состоит в том, что агенты запускаются не непосредственно для решения определенной задачи, а активизируются самостоятельно в какие-то моменты времени. В свою очередь понятие «мобильные агенты» подразумевает агентов, способных переместить своё выполнение на другие процессоры для последующего выполнения там [6].

Главная особенность, отличающая мобильные агенты от других типов программ, – это их способность перемещаться по сети, причем перемещение происходит по их собственной инициативе. По этой причине следует, что мобильными агентами нельзя считать ни сценарии в составе HTML-страниц, ни апплеты, и даже сериализуемые объекты, допускающие копирование по сети, не могут называться мобильными агентами. На определенном этапе своего выполнения агент принимает решение, что для него будет целесообразным сменить «место жительства». Для этого мобильный агент инициирует свое перемещение, и на узле назначения агент продолжает свою работу в том же состоянии, в котором он находился перед началом перемещения [7].

Для создания мобильных агентов подходят язык программирования, обеспечивающие реальную переносимость кода. Предпринималось множество попыток по реализации мобильных агентов на интерпретируемых языках, в частности на языке Tcl, но чаще всего попытки реализовать мобильный агент делаются с помощью языка Java. Но каким бы удачным не был выбор языка реализации агента, какие бы хорошие решения не предпринимал разработчик, мобильный агент не будет работать на том узле, где не осуществляется его поддержка. Иначе бы вся сеть очень скоро была бы наводнена вирусами, созданными по такому принципу. Перемещение мобильного агента возможно лишь на то устройство, где есть вся необходимая инфраструктура для поддержки агентов данного конкретного типа, которая иногда может быть реализована через достаточно сложные программы [7].

На сегодняшний день для маршрутизации пакетов по сети часто используется протокол динамической маршрутизации по состоянию каналов – протокол OSPF. Можно предположить, что вместо разработки нового протокола можно модернизировать существующий протокол OSPF. Но проблема в том, что написание для него мобильного агента без серьезных изменений его алгоритмов работы является очень сложной задачей. Все дело в том, что в своих таблицах, содержащих информацию о топологии сети, OSPF не содержит информации об альтернативных маршрутах, что делает отправку мобильных агентов с целью исследования сети и сбора служебной информации задачей труднореализуемой.

2.2 Протокол

Классический протокол AntNet − это адаптивный, распределенный алгоритм маршрутизации. Элементы AntNet, включают непременный атрибут любого маршрутизатора – таблицу маршрутизации.

Процедура поиска кратчайших путей в алгоритме AntNet использует те же принципы, что и колония муравьев при поиске пиши. Муравей на своем пути оставляет след из активных веществ – феромонов. Феромоновый след во внешней среде существует некоторое время, и муравьи, которые будут идти следом, с большей вероятностью предпочтут то направление, по которому до них прошли другие муравьи. Эта вероятность будет тем больше, чем большее количество муравьев прошло по этому пути. Таким образом, через некоторое время большая часть муравьев будет использовать один, наиболее близкий к оптимальному, маршрут.

Роль муравьев в протоколе AntNet выполняют активные агенты. Активный агент − это специальный пакет, который несет с собой статистику о состоянии пройденных сетевых каналов. Такими пакетами могут являться пакеты, называемые ANT_INIT. Пакеты ANT_INIT собирают во внутренний стек статистику о состоянии сети. Они не изменяют таблицы маршрутизации. При прохождении маршрутизаторов, пакеты ANT_INIT испытывают те же задержки, что и обыкновенные пакеты данных, поэтому собранная ими информация в большей мере соответствует действительному состоянию сети. Каждый маршрутизатор, реализующий протокол AntNet, с заданной периодичностью рассылает пакеты ANT_INIT в различные узлы сети, тем самым отслеживая ее состояние.

Работа предполагаемого роутера могла бы быть представлена следующим образом. На начальном этапе работы происходит инициализация сети, и все узлы сети настраивают свои таблицы маршрутизации с использованием протокола роевого интеллекта AntNet. Затем маршрутизатор приступает к началу передачи полезных данных – IP-пакетов. После того, как таймер времени рассылки пакетов ANT_INIT обнулится, маршрутизатор создает сообщение «ANT_INIT», так называемого «муравья», который используется в протоколе для определения RTT маршрута (round-trip time). RTT является ключевым параметром в определении оптимального маршрута.

После рассылки сообщений ANT_INIT маршрутизатор продолжает передачу полезных IP-пакетов по исходному маршруту, который был установлен ранее в таблице маршрутизации.

ANT_INIT рассылается по всем интерфейсам маршрутизатора, то есть с помощью широковещательной передачи. Пакет будет следовать до узла назначения не всегда оптимальными путями, но узлы сети, которые не ожидают прихода данного сообщения ANT_INIT, будут просто отбрасывать такой пакет. К узлам, отбрасывающим определенный пакет ANT_INIT, относятся узлы, которые уже получали данный пакет. Это будет возможно в виду того, что в процессе путешествия в ANT_INIT формируется особый стек, где фиксируется информация о посещенных узлах.

Дойдя до конечного узла, пакет ANT_INIT проделает весь путь в обратном порядке по пройденному ранее маршруту. Так как ANT_INIT хранит в своем стеке пройденные узлы, то узлы, на которых он не был, будут просто отбрасывать такие пакеты.

Исходный маршрутизатор, дождавшись ANT_INIT, получает необходимую информацию о времени прохождения маршрута. Имеющиеся данные можно интерпретировать как реальную метрику пути, учитывающую текущую конфигурацию и загрузку сети. Сведения, поступившие от различных пакетов ANT_INIT, позволяют достаточно точно сравнить стоимость прохождения пакета по разным маршрутам и, соответственно, оптимизировать таблицу маршрутизации.

Принцип определения маршрутов, применяемый в системе AntNet, обеспечивает движение пакетов по различным направлениям с учетом реальной загрузки линий. Данный алгоритм функционирования протокола AntNet порождает определенные трудности, так как следует учитывать резкое увеличение накладных расходов, связанных со сбором информации о сети. Так же увеличивается и объем данных о маршрутах – ведь пакет ANT_INIT движется к узлу назначения не всегда по кратчайшему пути. Но

несмотря на многократное повышение накладных расходов, данные, передаваемые AntNet, занимают малую часть пропускной способности линий связи, в особенности если учесть положительный эффект от использования протокола.

2.3 Граф работы протокола

Граф работы протокола маршрутизации, основанного на роевом интеллекте – протокола AntNet − может быть представлен следующим образом (рисунок 3).

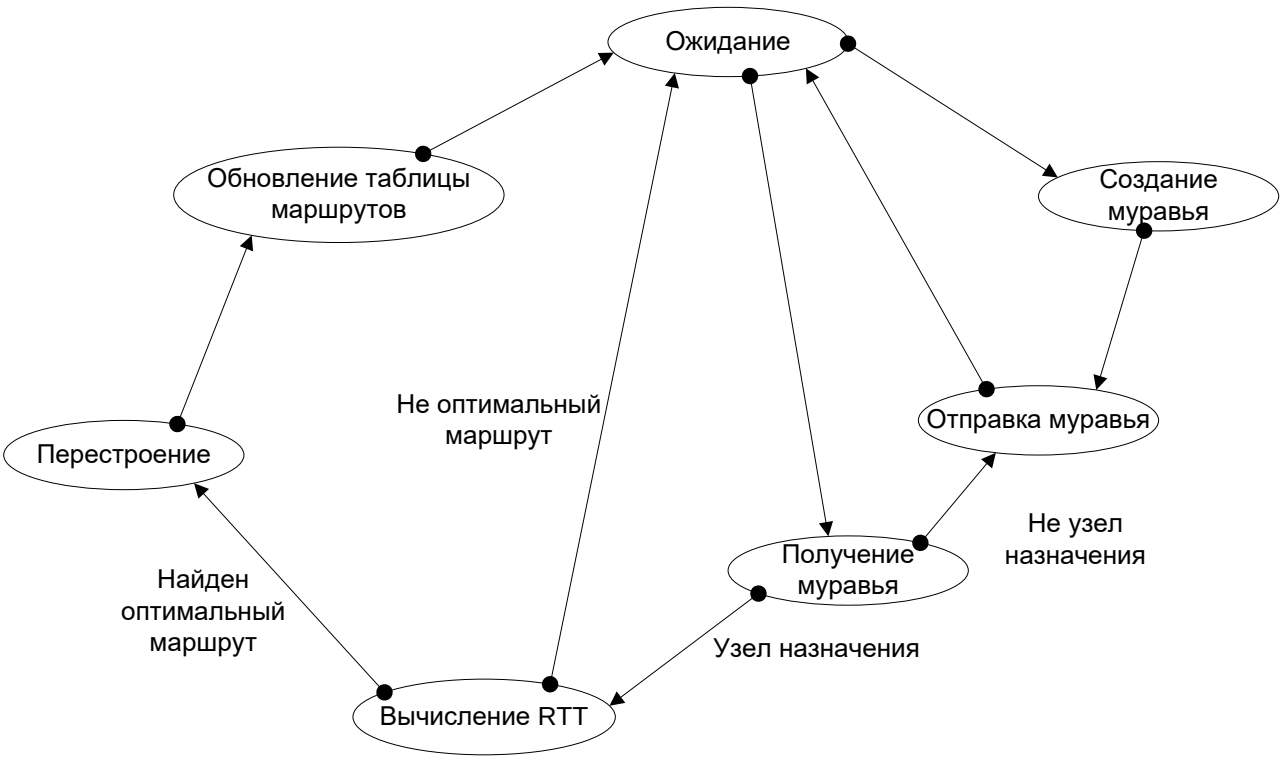

Рисунок 3 – Граф работы протокола AntNet

Работа протокола начинается в состоянии «Ожидание», то есть в состоянии ожидания событий. В состоянии «Ожидание» протокол может находиться бесконечно долго.

После некоторых внешних воздействий протокол переходит в состояние «Создание муравья». Под внешним воздействием понимается обнуление таймера отправки пакета ANT_INIT. В данном состоянии протокол генерирует пакет ANT_INIT, после чего протокол переходит в состояние «Отправка муравья» и производит рассылку пакетов ANT_INIT по всем интерфейсам узла. После рассылки всех пакетов ANT_INIT протокол возвращается в начальное состояние «Ожидание». Пакеты ANT_INIT рассылаются по сети, замеряя параметр RTT. После получения каждого ранее отправленного пакета ANT_INIT, протокол переходит в состояние «Получение муравья», после чего происходит определение, достиг ли полученный пакет узла назначения. Если пакет не достиг узла назначения, то происходит переход в состояние «Отправка муравья», после чего протокол возвращается в состояние «Ожидание». Если пакет достиг целевого узла назначения, то протокол переходит в состояние «Вычисление RTT», в котором происходит подсчет RTT пакета ANT_INIT с проделанного ранее маршрута. В случае если оптимального маршрута не было выявлено, протокол возвращается в состояние «Ожидание». Если же оптимальный маршрут был найден, то протокол переходит в состояние «Перестроение». В этом состоянии происходит изменение существующей информации о маршрутах, выбираются наиболее оптимальные маршруты следования. После протокол переходит в состояние «Обновление таблицы маршрутизации» и обновляет таблицу маршрутизации, после чего маршрут по умолчанию до целевого узла меняется, пакеты начинают передаваться по оптимальному маршруту, а протокол возвращается в начальное состояние «Ожидание». Таким образом, в зависимости от того как будет меняться загрузка сети будет меняться и таблица маршрутизации.

3 Реализация AntNet

3.1 Обзор среды OMNeT++

OMNeT++ − это расширяемый, модульный фреймворк, на основе компонентов библиотеки C++, используемый для построения моделей сети, представляет собой симулятор дискретных событий. В системе OMNeT++ заложена детальная реализация протоколов, начиная от сетевого уровня, возможность написания и подключения собственных модулей, развитый графический интерфейс [8].

Изменение состояния моделируемой системы происходит в дискретные моменты времени по списку будущих событий (future eventlist), отсортированных по времени. Событием может быть: начало передачи пакета, тайм-аут и т. п. События происходят на основе выполнения простых модулей (simple module). У такого модуля есть функции инициализации, обработки сообщения, действия и завершения работы [8].

Система INET Framework – это комплект модулей с открытым исходным кодом, позволяющих реалистично моделировать узлы и протоколы проводных и беспроводных сетей. Он включает модели различных протоколов Интернета: IP, IPv6, TCP, UDP, 802.11, Ethernet, PPP, MPLS с LDP и RSVP-TE signalling, OSPF версии 2 и ряд других. В комплект также входят различные реалистичные примеры использования этих протоколов [8].

Наивысший уровень абстракции в моделировании IP в INET Framework – это сеть, состоящая из IP-узлов. Узел может быть маршрутизатором или хостом. IP-узел отвечает компьютерному представлению стека протоколов Интернета. Модули, из которых он состоит, организованы так, как происходит обработка IP-дейтаграммы в операционных системах. Обязательным является модуль, отвечающий за сетевой уровень (реализующий обработку IP) и модуль

“сетевой интерфейс”. Дополнительно подключаются модули, реализующие протоколы транспортного уровня [8].

Помимо модуля INET, для OMNeT++ существует еще несколько модулей, которые не будут использованы в данной работе, так как в этом нет необходимости. Наименование этих модулей:

− MiXiN − является фреймворком OMNeT++. MiXiM создан для моделирования мобильных и беспроводных сетей (беспроводных сенсорных сетей, узкоспециализированных сетей, автомобильных сетей и т.д.). MiXiM концентрируется на нижних уровнях стека протоколов, а также предлагает детальные модели распространения радиоволн, оценки помех, встроенный радиопередатчик энергопотреблением и беспроводных протоколов MAC [9];

− Castalia − является симулятором работы беспроводных сенсорных сетей (Wireless Sensor Network, WSN), сетей BAN (Body Area Networks) и в целом сетей маломощных встраиваемых устройств [9].

3.2 OMNeT++ модель

Тестовая сеть была построена с использованием инструмента построения и моделирования работы сетей OMNeT++. Для проверки работоспособности протокола роевого интеллекта была построена следующая модель сети, представленная на рисунке 4.

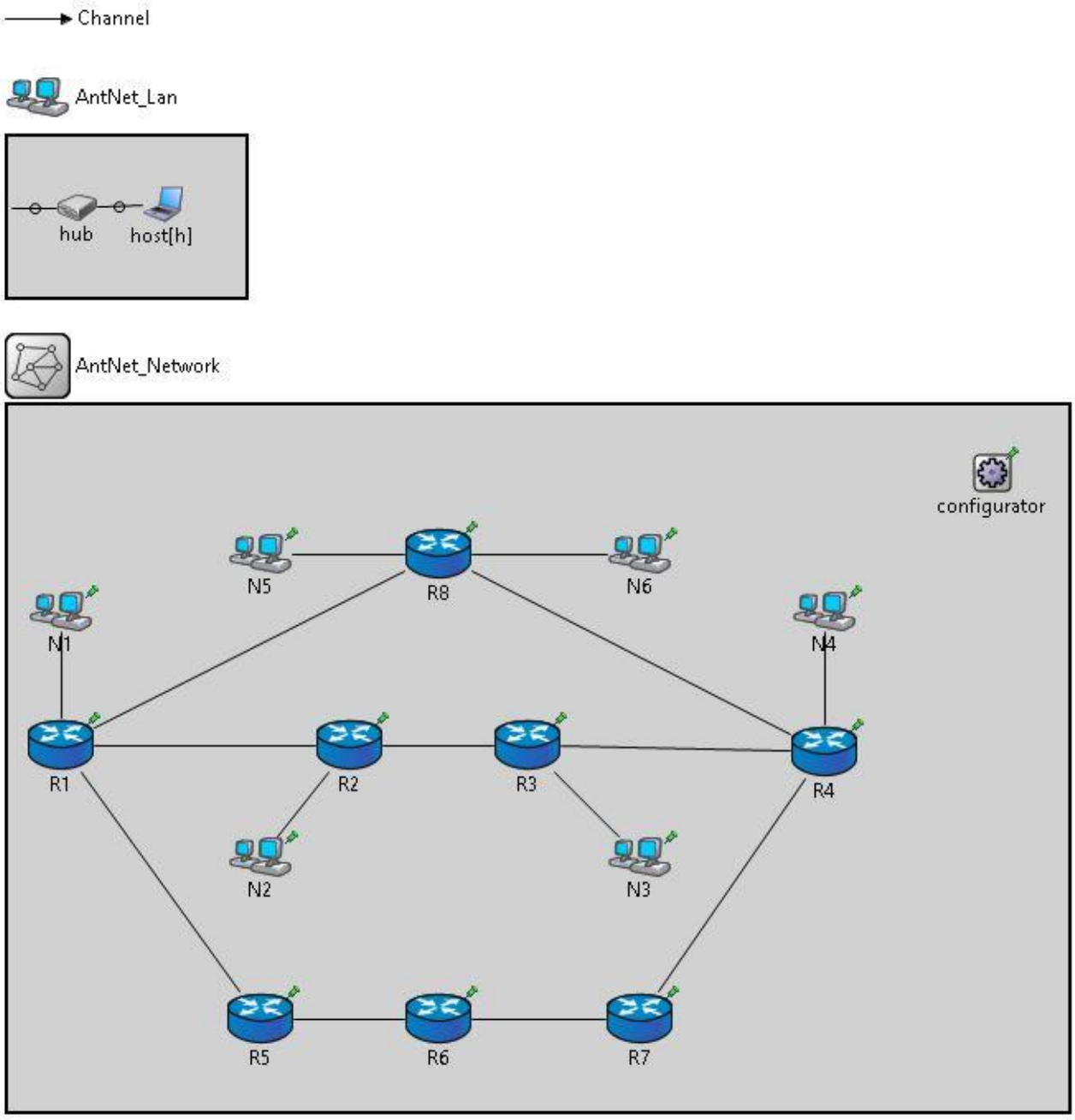

Рисунок 4 – Модель сети, построенная в OMNeT++

Сеть представляет собой топологию, состоящую из 8 ед. маршрутизаторов. Маршрутизаторы сети имеют названия «R*», − где символ * − номер маршрутизатора. Так же, в сети присутствуют узлы, представляющие собой генераторы трафика – это внутренние локальные сети. Внутренние локальные сети имеют обозначение «N*», − где символ *

так же представляет собой номер узла в полученной сети. Таких узлов в сети представлено 6 ед. Данные узлы необходимы для создания трафика, который будет передаваться по сети. Некоторые узлы созданы для создания паразитной нагрузки, чтоб продемонстрировать алгоритм перестроения на оптимальный маршрут передачи данных, заложенный в протокол AntNet.

Внутренние локальные сети (обозначение на рисунке «N*») имеют свою собственную топологию сети (рисунок 5).

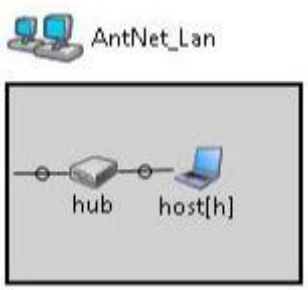

Рисунок 5 – Топология внутренних локальных сетей

Данная топология состоит из сетевого коммутатора (hub) и из некоторого количества хостов (host[h]), которые соединены с сетевым коммутатором каналами связи. Количество хостов розница от 1 до 2 ед.

Так же, в топологии представленной сети присутствует элемент с названием «configurator», который располагается в правом верхнему углу сети. В OMNeT++ данный модуль имеет название IPv4NetworkConfigurator. Этот модульнеобходим для выдачи маршрутизаторам основных настроек, необходимых для работы в сетях, поддерживающих технологию TCP/IP. Основными настройками, выдаваемые модулем IPv4NetworkConfigurator, являются IP-адреса узлов версии IPv4, а так же IP-адреса каждого интерфейса узла.

В ходе работы сети будет продемонстрирована инициализация сети и поиск оптимального маршрута для передачи пакетов от одного узла до другого.

Описанная выше топология сети была построена с целью проверки работы алгоритмов поиска оптимального маршрута протоколом AntNet и последующим перестроением на оптимальный маршрут передачи данных в сети.

3.3 Алгоритм протокола

3.3.1 Алгоритм работы протокола AntNet

Работу протокола роевого интеллекта можно условно разбить на два основных этапа:

1. инициализация сети;
2. поддержание оптимальных маршрутов;
3. обработка сбоев.

Этап «Инициализация сети» подразумевает начальную настройку сети при ее запуске. При запуске сети необходимо построить начальную маршрутизацию для возможности передачи данных по сети. Для этого необходимо, чтобы каждый маршрутизатор получил начальные сведения о состоянии сети, и на основе этих данных построил свою таблицу маршрутизации.

Следующим этапом работы протокола роевого интеллекта является этап, условно названный «Поддержание оптимальных маршрутов». Основная идея данного этапа работы заключается в том, в процессе работы и функционирования сети загрузка каналов связи будет меняться, вследствие чего передача данных через такие каналы связи может быть не самой оптимальной, так как полезные пакеты данных будут вставать в очередь на

передачу, тем самым они и сами будут только увеличивать эту очередь передачи. Поэтому данный этап работы сети необходим для поддержания передачи трафика сети по оптимальным маршрутам связи. По сути, данный этап реализует метод балансировки нагрузки, основная идея которого заключается в распределении заданий между сетевыми устройствами с целью оптимизации использования ресурсов, сокращения времени обслуживания запросов, а также обеспечения отказоустойчивости (резервирования).

Этап, называемый «Обработка сбоев» является не менее важным в ходе функционировании протокола роевого интеллекта, чем и предыдущие этапы работы протокола. Данный этап ориентирован на случай, когда в работе сети происходит сбой в работе и какой-то из узлов выходит из строя, вследствие чего канал передачи данных тоже выходит из строя и логическая топология сети рушится. В таком случае логическую топологию сети необходимо перестраивать, чтобы передача данных по сети могла и дальше происходить без потери полезных пакетов.

3.3.2 Инициализация сети

Алгоритм работы протокола на этапе инициализации сети может быть представлен в виде следующего алгоритма:

− администратор сети вручную производит выбор корневого (главного) маршрутизатора в сети. Выбор данного маршрутизатора необходим ввиду того, что с него будет начинаться рассылка пакетов ANT_INIT;

− затем, так как таблицы маршрутизации роутеров пустые, роутерам необходимо установить начальные связи с соседними маршрутизаторами. Для этого каждый маршрутизатор производит широковещательную рассылку по всем интерфейсам пакета ANT_HELLO. Предварительно, перед отправкой пакета ANT_HELLO, каждый маршрутизатор помечает время отправки

каждого из пакетов, для дальнейшего вычисления времени нахождения пакета в пути;

− целевые маршрутизаторы, получая пакет ANT_HELLO, считывают, откуда был получен данный пакет (с какого интерфейса маршрутизатора), затем производят уничтожение данного пакета и создание его зеркальной копии, отправляя копию по тому же интерфейсу, по которому и было произведено получение данного пакета;

− маршрутизатор, получая зеркальную копию своего пакета ANT_HELLO, фиксирует время его получения, которое сравнивает со временем отправки, полученная разность этих двух времен и отражает время нахождения пакета в пути (параметр RTT). RTT хранится в отдельной таблице, а RTT соседних маршрутизаторов не удаляется из таблицы до того момента, пока между устройствами существует канал логической связи для передачи сетевых данных. После вычисления RTT для каждого маршрута происходит обновление таблицы маршрутизации роутера, и все маршруты до соседних маршрутизаторов записываются в таблицу маршрутизации роутера. После описанных манипуляций начинается следующий этап работы протокола роевого интеллекта;

− корневой маршрутизатор производит широковещательную рассылку пакетов ANT_INIT по всем интерфейсам, фиксируя при этом время отправки каждого пакета. Каждый из отправленных пакетов имеет свой уникальный идентификатор, который необходим для того, чтоб маршрутизатор мог различать отправленные пакеты;

− соседи, получая пакет ANT_INIT, считывают, откуда был получен данный пакет (с какого интерфейса маршрутизатора), затем производят уничтожение данного пакета и создание его зеркальной копии, или пакета ANT_ECHO, отправляя копию по тому же интерфейсу, по которому и было произведено получение данного пакета. В то же время, маршрутизатор, получивший пакет ANT_INIT, производит его дальнейшую отправку по всем

своим сетевым интерфейсам, кроме того интерфейса, с которого был получен пакет ANT_INIT. Предварительно, маршрутизатор вписывается перед отправкой продолжения пакетов ANT_INIT и зеркального пакета ANT_ECHO, фиксируя тем самым тот факт, что полученный этим роутером пакет ANT_INIT проходил именно через этот маршрутизатор. Данная мера необходима ввиду того, чтобы в дальнейшем избежать появления закольцованного трафика, так маршрутизатор, получая широковещательные пакеты ANT_INIT будет отбрасывать те пакеты, которые уже приходили на данный маршрутизатор, не отправляя пакеты ANT_INIT и ANT_ECHO далее по сети;

− пакет ANT_ECHO проделывает весь путь в обратном направлении. Это возможно в связи с наличием в теле пакета специального стека, в котором по порядку хранится каждый пройденный пакетом ANT_INIT маршрутизатор. Маршрутизатор, получая пакет ANT_ECHO, производит анализ стека, и принимает решение по какому интерфейсу отправить пакет ANT_ECHO далее;

− корневой маршрутизатор, получая пакет ANT_ECHO, фиксирует время его получения, которое сравнивает со временем отправки, вычисляя параметр RTT. Находя маршрут до ранее неизвестного маршрутизатора, целевой роутер производит обновление таблицы маршрутизации, вписывая новый маршрут в таблицу. Находя маршруты до уже известных маршрутизаторов, происходит сравнение RTT маршрута, используемого по умолчанию, содержащегося в таблице маршрутизации, и нового маршрута из пакета ANT_ECHO. Если маршрут, используемый по умолчанию, имеет меньший RTT, то маршрут по умолчанию сохраняется, иначе происходит корректировка таблицы маршрутизации и маршрутом по умолчанию до целевого узла становится новый маршрут, полученный из пакета ANT_ECHO;

− пакеты ANT_INIT будут рассылаться до тех пор, пока не достигнут всех маршрутизаторов по каждому имеющемуся в сети маршруту, без образования петель трафика;

− каждый маршрутизатор, не являющийся корневым, получая свои первые два пакета ANT_INIT, помимо записи в них пометки о прохождении этого целевого роутера, дополнительно вписывают свою уникальную метку. Эта метка необходима для того, чтобы получая зеркальные пакеты, которые будут хранить в себе эту метку, маршрутизатор мог отличить эти пакеты от остальных зеркальных пакетов ANT_ECHO. Помеченные пакеты будут использоваться для вычисления RTT маршрутов и обновления таблицы маршрутизации на этом целевом роутере. Перед отправкой помеченного пакета ANT_INIT, маршрутизатор засекает время его отправки и сопоставляет с уникальной меткой. Метки для первого и второго пакета ANT_INIT будут разные, что бы различать зеркальные пакеты и сопоставлять их RTT;

− получая помеченный пакет ANT_ECHO маршрутизатор, как и корневой маршрутизатор, высчитывает параметр RTT маршрута и либо сохраняет маршрут, который используется по умолчанию, либо происходит корректировка таблицы маршрутизации и в таблицу вписывается новый маршрут;

− если в сети имеется маршрутизатор, который имеет только один сетевой интерфейс, то такой маршрутизатор строит свою таблицу маршрутизации через соседствующий с ним роутер, то есть запрашивает таблицу маршрутизации соседнего маршрутизатора отправкой пакета ANT_DB, предварительно зафиксировав время отправки пакета ANT_DB. Отправка пакета ANT_DB происходит после того, как данный роутер получит пакет ANT_INIT, но не сможет ответить на него последующей отправкой следующего пакета ANT_INIT. Отправка пакета ANT_DB происходит через минуту после получения ANT_INIT. Данное время выбрано из расчета, что бы соседний маршрутизатор успел построить свою таблицу маршрутизации, и таким образом, отправил актуальную таблицу маршрутизации;

− целевой маршрутизатор, получая пакет ANT_DB от своего соседа, производит отправку всей своей таблицы маршрутизации и таблицы RTT назад роутеру, подавшему запрос, отправляя ему пакет ANT_DB_ECHO;

− маршрутизатор, получая пакет ANT_DB_ECHO, фиксирует время прибытия пакета, производя вычисление параметра RTT. Затем маршрутизатор производит обновление таблицы маршрутизации, а RTT каждого маршрута высчитывается по принципу прибавления к каждой ячейке полученной от соседа таблицы RTT собственного параметра RTT, получая, таким образом, реальное для данного маршрутизатора значение каждого показателя RTT для каждого маршрута.

Этап инициализации сети будет происходить до тех пор, пока каждый маршрутизатор полностью не заполнит свою таблицу маршрутизации маршрутами до всех узлов, имеющихся в сети.

3.3.3 Поддержание оптимальных маршрутов

Алгоритм работы протокола на этапе поддержания таблицы оптимальных маршрутов для каждого маршрутизатора может быть представлен в виде следующего алгоритма:

− у протокола есть таймер, настроенный на 10 минут. Таймер необходим, чтобы каждые 10 минут производит проверку маршрутов по факту наличия наиболее оптимального маршрута, либо установления факта того, что используемый маршрут передачи данных является наиболее оптимальным. Такое время таймера выбрано из расчета, что данное время будет являться наиболее оптимальным. С данной периодичностью алгоритм будет часто перестраивать таблицу на наиболее оптимальный маршрут передачи данных, при этом это не будет происходить слишком часто, создавая тем самым большую паразитную нагрузку на сеть;

− как только таймер корневого маршрутизатора зафиксирует факт того, что прошло 10 минут, маршрутизатор рассылает по всем своим физическим интерфейсам пакеты ANT_INIT, узлами назначения которых будут являться все узлы сети, так как ANT_INIT будет передаваться широковещательно по всем интерфейсам роутера. Перед отправкой пакетов ANT_INIT маршрутизатором будет зафиксировано время их отправки, для последующего вычисления RTT, а так же каждому пакету будет присвоен уникальный идентификатор, для того, чтобы различать эти пакеты;

− целевые маршрутизаторы, не являющиеся корневым, получая пакет ANT_INIT, считывают, откуда был получен данный пакет (с какого интерфейса маршрутизатора), затем производят уничтожение данного пакета и создание его зеркальной копии, или пакета ANT_ECHO, отправляя копию по тому же интерфейсу, по которому и было произведено получение данного пакета. В то же время, маршрутизатор, получивший пакет ANT_INIT, производит его дальнейшую отправку по всем своим сетевым интерфейсам, кроме того интерфейса, с которого был получен пакет ANT_INIT. Предварительно, маршрутизатор вписывается перед отправкой продолжения пакетов ANT_INIT и зеркального пакета ANT_ECHO, фиксируя тем самым тот факт, что полученный этим роутером пакет ANT_INIT проходил именно через этот маршрутизатор. Данная мера необходима ввиду того, чтоб в дальнейшем избежать появления закольцованного трафика, так маршрутизатор, получая широковещательные пакеты ANT_INIT будет отбрасывать те пакеты, которые уже приходили на данный маршрутизатор, не отправляя пакеты ANT_INIT и ANT_ECHO далее по сети;

− пакет ANT_ECHO проделывает весь путь в обратном направлении. Это возможно в связи с наличием в теле пакета специального стека, в котором по порядку хранится каждый пройденный пакетом ANT_INIT маршрутизатор. Маршрутизатор, получая пакет ANT_ECHO, производит анализ стека, и принимает решение по какому интерфейсу отправить пакет ANT_ECHO далее,

чтобы он проделал весь путь в обратном направлении, которое составил пакет ANT_INIT;

− корневой маршрутизатор, получая пакет ANT_ECHO, фиксирует время его получения, которое сравнивает со временем отправки, вычисляя параметр RTT. Находя маршрут до ранее неизвестного маршрутизатора, целевой роутер производит обновление таблицы маршрутизации, вписывая новый маршрут в таблицу. Находя маршруты до уже известных маршрутизаторов, происходит сравнение RTT маршрута, используемого по умолчанию, содержащегося в таблице маршрутизации, и нового маршрута из пакета ANT_ECHO. Если маршрут, используемый по умолчанию, имеет меньший RTT, то маршрут по умолчанию сохраняется, иначе происходит корректировка таблицы маршрутизации и маршрутом по умолчанию до целевого узла становится новый маршрут, полученный из пакета ANT_ECHO;

− пакеты ANT_INIT будут рассылаться до тех пор, пока не достигнут всех маршрутизаторов по каждому имеющемуся в сети маршруту, без образования петель трафика;

− каждый маршрутизатор, не являющийся корневым, получая свои первые два пакета ANT_INIT, помимо записи в них пометки о прохождении этого целевого роутера, дополнительно вписывают свою уникальную метку. Эта метка необходима для того, чтобы получая зеркальные пакеты, которые будут хранить в себе эту метку, маршрутизатор мог отличить эти пакеты от остальных зеркальных пакетов ANT_ECHO. Помеченные пакеты будут использоваться для вычисления RTT маршрутов и обновления таблицы маршрутизации на этом целевом роутере. Перед отправкой помеченного пакета ANT_INIT, маршрутизатор засекает время его отправки и сопоставляет с уникальной меткой. Метки для первого и второго пакета ANT_INIT будут разные, что бы различать зеркальные пакеты и сопоставлять их RTT;

− получая помеченный пакет ANT_ECHO маршрутизатор, как и корневой маршрутизатор, высчитывает параметр RTT маршрута и либо сохраняет

маршрут, который используется по умолчанию, либо происходит корректировка таблицы маршрутизации и в таблицу вписывается новый маршрут;

− если в сети имеется маршрутизатор, который имеет только один сетевой интерфейс, то такой маршрутизатор строит свою таблицу маршрутизации через соседствующий с ним роутер, то есть запрашивает таблицу маршрутизации соседнего маршрутизатора отправкой пакета ANT_DB, предварительно зафиксировав время отправки пакета ANT_DB. Отправка пакета ANT_DB происходит после того, как данный роутер получит пакет ANT_INIT, но не сможет ответить на него последующей отправкой следующего пакета ANT_INIT. Отправка пакета ANT_DB происходит через минуту после получения ANT_INIT. Данное время выбрано из расчета, что бы соседний маршрутизатор успел построить свою таблицу маршрутизации, и таким образом, отправил актуальную таблицу маршрутизации;

− целевой маршрутизатор, получая пакет ANT_DB от своего соседа, производит отправку всей своей таблицы маршрутизации и таблицы RTT назад роутеру, подавшему запрос, отправляя ему пакет ANT_DB_ECHO;

− маршрутизатор, получая пакет ANT_DB_ECHO, фиксирует время прибытия пакета, производя вычисление параметра RTT. Затем маршрутизатор производит обновление таблицы маршрутизации, а RTT каждого маршрута высчитывается по принципу прибавления к каждой ячейке полученной от соседа таблицы RTT собственного параметра RTT, получая, таким образом, реальное для данного маршрутизатора значение каждого показателя RTT для каждого маршрута.

Как только каждый маршрутизатор получит все пакеты ANT_ECHO, которые ему предназначались, этап поддержания оптимальных маршрутов на маршрутизаторе завершится.

Следует отметить, что на время поиска оптимальных маршрутов маршрутизаторами сети, передача полезного трафика в сети не

приостанавливается. Алгоритм поддержание таблицы оптимальных маршрутов работает параллельно с передачей трафика по сети. Только таким образом можно отследить загрузку каналов.

3.3.4 Обработка сбоев

Алгоритм работы протокола на этапе обработки сбоев в ходе функционировании сети может быть представлен в виде следующего алгоритма:

− каждый раз, перед началом передачи данных от узла источника к узлу приемнику, происходит процедура установки соединения между узлами через транспортный протокол TCP. В следствие данной процедуры, если два узла не могут установить соединение после 6 попыток (что соответствует минуте астрономического времени), то маршрутизатор удаляет этот маршрут из таблицы маршрутизации;

− если же соединение будет разорвано в процессе передачи данных, то узел источник сможет отследить данный сбой после того, как узел назначения не отправит пакет, подтверждающий прием данных. В качестве пакета, подтверждающего факт приема данных, в протоколе TCP являются эхо пакеты. Если целевой узел назначения более 6 раз подряд не отправит факт приема данных, от маршрутизатор автоматически удаляет данный маршрут из своей таблицы маршрутизации;

− затем, так как маршрут по умолчанию был удален из таблицы маршрутизации маршрутизатора источника, то данный маршрутизатор производит отправку корневому роутеру пакета ANT_REINIT;

− корневой маршрутизатор, получая пакет ANT_REINIT, инициализирует проверку и перестроение сети. Для этого корневой маршрутизатор производит широковещательную рассылку пакетов ANT_INIT по всем интерфейсам, фиксируя при этом время отправки каждого пакета. Каждый из отправленных

пакетов имеет свой уникальный идентификатор, который необходим для того, чтоб маршрутизатор мог различать отправленные пакеты;

− соседи, получая пакет ANT_INIT, считывают, откуда был получен данный пакет (с какого интерфейса маршрутизатора), затем производят уничтожение данного пакета и создание его зеркальной копии, или пакета ANT_ECHO, отправляя копию по тому же интерфейсу, по которому и было произведено получение данного пакета. В то же время, маршрутизатор, получивший пакет ANT_INIT, производит его дальнейшую отправку по всем своим сетевым интерфейсам, кроме того интерфейса, с которого был получен пакет ANT_INIT. Предварительно, маршрутизатор вписывается перед отправкой продолжения пакетов ANT_INIT и зеркального пакета ANT_ECHO, фиксируя тем самым тот факт, что полученный этим роутером пакет ANT_INIT проходил именно через этот маршрутизатор. Данная мера необходима ввиду того, чтоб в дальнейшем избежать появления закольцованного трафика, так маршрутизатор, получая широковещательные пакеты ANT_INIT будет отбрасывать те пакеты, которые уже приходили на данный маршрутизатор, не отправляя пакеты ANT_INIT и ANT_ECHO далее по сети;

− пакет ANT_ECHO проделывает весь путь в обратном направлении. Это возможно в связи с наличием в теле пакета специального стека, в котором по порядку хранится каждый пройденный пакетом ANT_INIT маршрутизатор. Маршрутизатор, получая пакет ANT_ECHO, производит анализ стека, и принимает решение по какому интерфейсу отправить пакет ANT_ECHO далее;

− корневой маршрутизатор, получая пакет ANT_ECHO, фиксирует время его получения, которое сравнивает со временем отправки, вычисляя параметр RTT. Находя маршрут до ранее неизвестного маршрутизатора, целевой роутер производит обновление таблицы маршрутизации, вписывая новый маршрут в таблицу. Находя маршруты до уже известных маршрутизаторов, происходит сравнение RTT маршрута, используемого по умолчанию, содержащегося в таблице маршрутизации, и нового маршрута из пакета ANT_ECHO. Если

маршрут, используемый по умолчанию, имеет меньший RTT, то маршрут по умолчанию сохраняется, иначе происходит корректировка таблицы маршрутизации и маршрутом по умолчанию до целевого узла становится новый маршрут, полученный из пакета ANT_ECHO;

− пакеты ANT_INIT будут рассылаться до тех пор, пока не достигнут всех маршрутизаторов по каждому имеющемуся в сети маршруту, без образования петель трафика;

− каждый маршрутизатор, не являющийся корневым, получая свои первые два пакета ANT_INIT, помимо записи в них пометки о прохождении этого целевого роутера, дополнительно вписывают свою уникальную метку. Эта метка необходима для того, чтобы получая зеркальные пакеты, которые будут хранить в себе эту метку, маршрутизатор мог отличить эти пакеты от остальных зеркальных пакетов ANT_ECHO. Помеченные пакеты будут использоваться для вычисления RTT маршрутов и обновления таблицы маршрутизации на этом целевом роутере. Перед отправкой помеченного пакета ANT_INIT, маршрутизатор засекает время его отправки и сопоставляет с уникальной меткой. Метки для первого и второго пакета ANT_INIT будут разные, что бы различать зеркальные пакеты и сопоставлять их RTT;

− получая помеченный пакет ANT_ECHO маршрутизатор, как и корневой маршрутизатор, высчитывает параметр RTT маршрута и либо сохраняет маршрут, который используется по умолчанию, либо происходит корректировка таблицы маршрутизации и в таблицу вписывается новый маршрут;

− если в сети имеется маршрутизатор, который имеет только один сетевой интерфейс, то такой маршрутизатор строит свою таблицу маршрутизации через соседствующий с ним роутер, то есть запрашивает таблицу маршрутизации соседнего маршрутизатора отправкой пакета ANT_DB, предварительно зафиксировав время отправки пакета ANT_DB. Отправка пакета ANT_DB происходит после того, как данный роутер получит пакет

ANT_INIT, но не сможет ответить на него последующей отправкой следующего пакета ANT_INIT. Отправка пакета ANT_DB происходит через минуту после получения ANT_INIT. Данное время выбрано из расчета, что бы соседний маршрутизатор успел построить свою таблицу маршрутизации, и таким образом, отправил актуальную таблицу маршрутизации;

− целевой маршрутизатор, получая пакет ANT_DB от своего соседа, производит отправку всей своей таблицы маршрутизации и таблицы RTT назад роутеру, подавшему запрос, отправляя ему пакет ANT_DB_ECHO;

− маршрутизатор, получая пакет ANT_DB_ECHO, фиксирует время прибытия пакета, производя вычисление параметра RTT. Затем маршрутизатор производит обновление таблицы маршрутизации, а RTT каждого маршрута высчитывается по принципу прибавления к каждой ячейке полученной от соседа таблицы RTT собственного параметра RTT, получая, таким образом, реальное для данного маршрутизатора значение каждого показателя RTT для каждого маршрута.

Корректировка таблиц маршрутизации на роутерах будет происходить до тех пор, пока каждый маршрутизатор полностью не обновит свою таблицу маршрутизации актуальными маршрутами до каждого узла.

Таким образом, происходит обработка сбоев в разработанном протоколе.

3.4 Структуры пакетов

Структура пакета ANT_HELLO представлена в таблице 2.

Таблица 2 – Структура пакета ANT_HELLO

Бит	0 − 15	16 − 31
0	Тип пакета	Идентификатор

Поле «Тип пакета» содержит тип ANT-пакета. Для пакетов ANT_HELLO данное поле = 1.

Поле «Идентификатор» содержит уникальный идентификатор который был присвоен пакету маршрутизатором источником.

Структура пакета ANT_INIT представлена в таблице 3.

Таблица 3 – Структура пакета ANT_INIT

Бит	0 − 15	16 − 31
0	Тип пакета	Идентификатор
32	Метки некорневых узлов	
192	Стек пройденных узлов	
…	...	

Для пакетов ANT_ INIT данное поле = 2.

Поле «Идентификатор» содержит уникальный идентификатор, который был присвоен пакету корневым маршрутизатором.

Поле «Метки некорневых узлов» содержит уникальные метки. В данное поле промежуточный узел прописывает свою уникальную метку для первых двух пакетов ANT_INIT.

Поле «Стек пройденных узлов» содержит все узлы, через которые прошёл пакет.

Структура пакета ANT_ECHO представлена в таблице 4.

Таблица 4 – Структура пакета ANT_ECHO

Бит	0 − 15	16 − 31
0	Тип пакета	Идентификатор
32	Метки некорневых узлов	
192	Стек пройденных узлов	
…	...	

Для пакетов ANT_ECHO данное поле = 3.

Структура пакета ANT_DB представлена в таблице 5.

Таблица 5 – Структура пакета ANT_DB

Бит	0 − 15	16 − 31
0	Тип пакета	Идентификатор

Для пакетов ANT_DB данное поле = 4.

Структура пакета ANT_DB_ECHO представлена в таблице 6.

Таблица 6 – Структура пакета ANT_DB_ECHO

Бит	0 − 15	16 − 31
0	Тип пакета	Идентификатор
32	Данные	
…	...	

Для пакетов ANT_DB_ECHO данное поле = 5.

Поле «Данные» содержит таблицу маршрутизации и таблицу RTT отправителя пакета ANT_DB_ECHO.

Структура пакета ANT_REINIT представлена в таблице 7.

Таблица 7 – Структура пакета ANT_REINIT

Бит	0 − 15	16 − 31
0	Тип пакета	Идентификатор

Для пакетов ANT_DB данное поле = 6.

4 Моделирование

4.1 Результаты эксперимента

На рисунке 6 представлена тестовая сеть, созданная в системе OMNeT++.

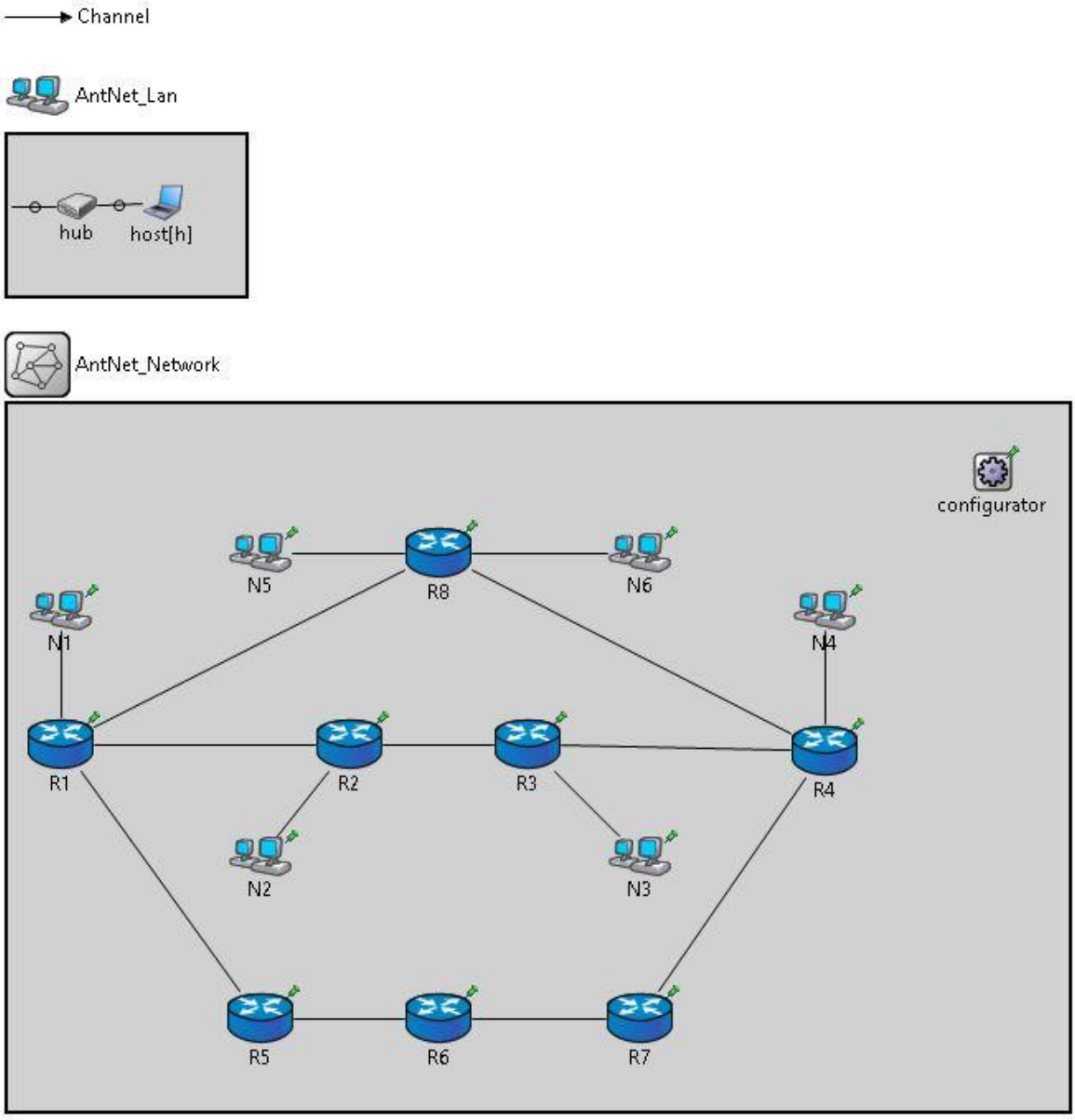

Рисунок 6 – Модель сети, построенная в OMNeT++

Внутренняя структура роутера, использующего в качестве протокола маршрутизации протокол AntNet, представлена на рисунке 7.

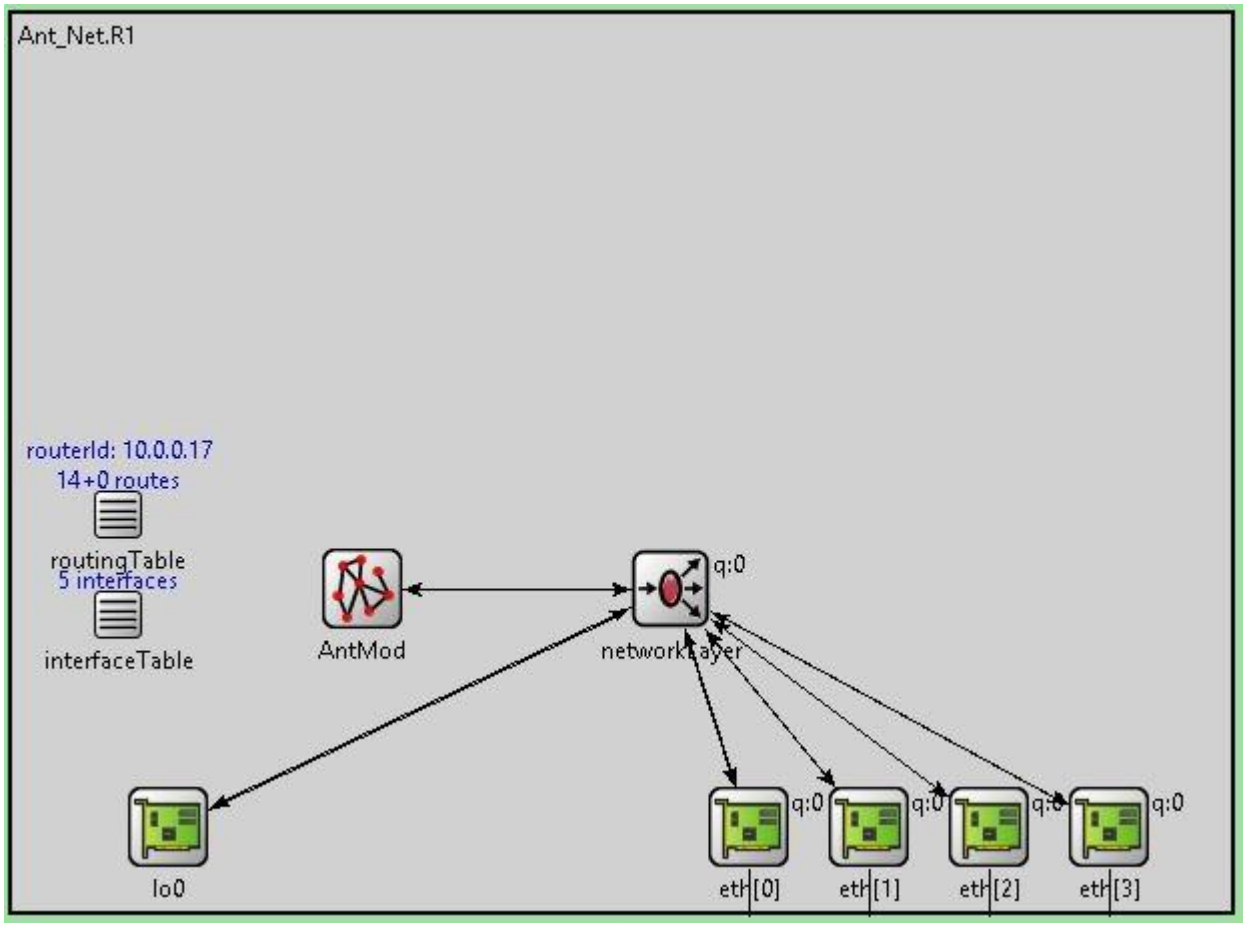

Рисунок 7 – Структура роутера R1

Из представленной на рисунке структуры видно, что роутер R1 состоит их нескольких сетевых интерфейсов, таблицы маршрутизации и модулей, среди которых есть модуль AntMod, реализующий работу протокола AntNet на данном роутере. Таблица маршрутизации включает 14 записей. Таблица интерфейсов включает интерфейс loopback и 4 интерфейса, соединяющих роутер R1 с другими элементами сети.

По окончанию процесса симуляции работы сети в папке results проекта AntProject был получен файл MyApp-0.vec. Данный файл был использован для

получения графика, на котором показан вектор пропускной способности роутера R1. График отображен на рисунке 8.

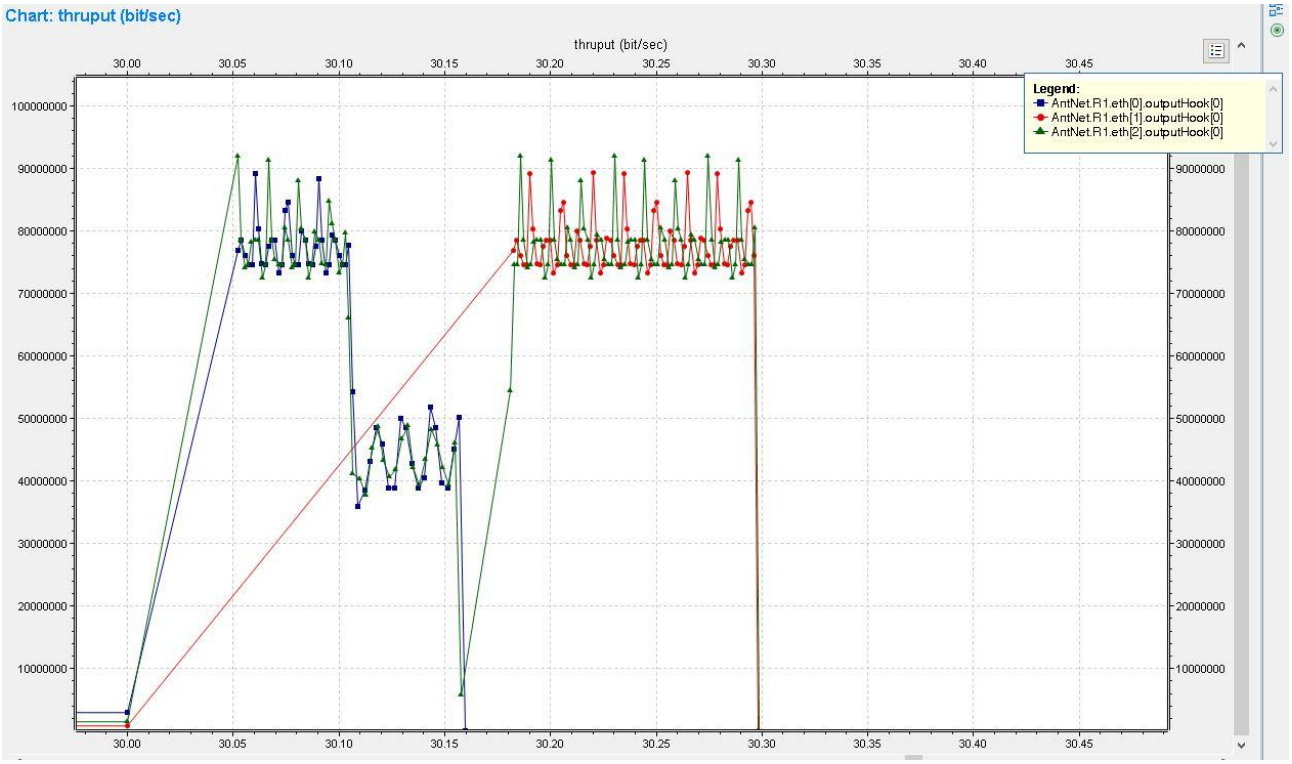

Рисунок 8 – График, полученный в результате моделирования AntNet

По оси абсцисс отложено время симуляции, а по оси ординат – величина пропускной способности.

В специальном файле ASConfig.xml, который принадлежит проекту сети, были заданы начальные параметры маршрутизации для маршрутизатора R1.

В момент времени 30 сек. происходит инициализация и начало передачи полезных сообщений с сети N1 на N4. Перед запуском сети, маршрутизация на роутере R1 была настроена так, что маршрут от сети N1 на N4 будет проходить через роутеры R1-R2-R3-R4. Модуль AntMod маршрутизатора R1, который является главным, отправит муравьев и тем самым обнаружит, что RTT маршрута по умолчанию является больше, чем на альтернативном маршруте, проходящем через роутеры R1-R8-R4. После чего

будет изменена таблица маршрутизации и пакеты пойдут по маршруту N1-R1-R8-R4-N4. На графике в момент времени 30,1676 происходит обрыв связи вследствие удаления действующего маршрута и добавления нового. В момент времени 30.1863 завершается процесс обновления таблицы маршрутизации на роутере R1, вследствие чего данные на маршрутизаторе R1 начинают передаваться через шлюз eth[1], который соединен с роутером R8. Пропускная способность каналов роутера R1 вновь используется в полной величине.

ЗАКЛЮЧЕНИЕ

В данной работе был разработан протокол маршрутизации на основе алгоритма роевого интеллекта – протокол AntNet. Для данного протокола был сформулирован алгоритм работы протокола. На основе алгоритма работы был сформирован граф работы протокола. Описаны структуры пакетов. Данный протокол был промоделирован в среде моделирования OMNeT++. Основанием разработки данного протокола послужила проблема того, что современные протоколы маршрутизации неспособны эффективно решать проблему отслеживания загруженности каналов связи и производить последующую перестройку таблицы маршрутизации.

С помощью библиотеки INET Framework для OMNeT++ был реализован модуль, обеспечивающий работу протокола AntNet.

Моделирование протокола роевого интеллекта было выполнено на разработанной для этого тестовой сети. На основании полученных результатов были сделаны выводы, что протокол AntNet в определенный момент времени отследил, что на используемом маршруте передачи данных время следования пакетов больше чем на другом маршруте. Таким образом, протокол отследил использование неоптимального маршрута передачи данных. После чего таблица маршрутизации была перестроена, и данные стали передаваться по другому маршруту.

В процессе моделирования AntNet были сделаны выводы, что протокол может быть использован в исследованиях по повышению производительности сети при больших нагрузках за счет использования алгоритмов, с помощью которых замеряется время следования пакетов по сети и происходит перенаправление трафика по маршруту с меньшим показателем RTT.

В результате выполнения данной работы был разработан протокол маршрутизации на основе алгоритма роевого интеллекта. Для достижения

эффективности в работе протокола AntNet требуются дополнительные исследования.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Система менеджмента качества. Общие требования к построению, изложению и оформлению документов учебной деятельности. СТО 4.2-07-2014. Красноярск.: ИПЦ СФУ, 2014. – 60 с.

2 Сайт о программировании Realcoding.Net [Электронный ресурс] – Режим доступа: http://www.realcoding.net/articles/vvedenie-v-protokol-ospf.html

3 Электронный журнал ITC [Электронный ресурс] – Режим доступа: http://itc.ua/articles/razgovor_o_marshrutizacii_ne_okonchen_25136/

4 Электронный портал университета [Электронный ресурс] – Режим доступа: https://lektsii.org/5-64716.html

5 Электронная энциклопедия [Электронный ресурс] – Режим доступа: http://dic.academic.ru/dic.nsf/ruwiki/1605981

6 Динамический алгоритм маршрутизации на основе агентных технологий, Солдатова В.А. Кафедра ПМИ ДонНТУ [Электронный ресурс] – Режим доступа: http://masters.donntu.org/2005/fvti/soldatova/library/soldatova.htm

7 Ant Colony Optimization and its Application to Adaptive Routing in Telecommunication Networks : дис. д-ра физ.-мат. наук / Gianni Di Caro. – Bruxelles, 2004 – 121 с.

8 Электронная энциклопедия [Электронный ресурс] – Режим доступа: http://rain.ifmo.ru/cat/data/theory/unsorted/ant-algo-2006/article.pdf

9 AntNet: Distributed Stigmergetic Control for Communications Networks / Gianni Di Caro, Marco Dorigo // Journal of Articial Intelligence Research 9 : сб. науч. ст. / IRIDIA, Universite Libre de Bruxelles, 1998. − С. 317–365.

10 Электронная энциклопедия [Электронный ресурс] – Режим доступа: https://studfiles.net/preview/933652/page:2/

11 Электронный журнал [Электронный ресурс] – Режим доступа: http://fb.ru/article/369611/nastroyka-staticheskoy-marshrutizatsii

12 Водолазский И. А. Роевой интеллект и его наиболее распространённые методы реализации / Водолазский И. А., Егоров А. С., Краснов А. В. // Молодой ученый. − 2017. − №4. − С. 147–153.

13 Электронный архив Донецкого национального технического университета [Электронный ресурс] – Режим доступа: http://ea.donntu.org:8080/bitstream/123456789/5592/1/p_169.pdf

14 Ладыженский Ю. В. Моделирование алгоритмов маршрутизации в сетях на кристалле / Ладыженский Ю. В., Мирецкая В. А. – Донецкий национальный технический университет, 2017. – №4. – С. 79–86.

15 Штовба С. Д. Муравьиные алгоритмы / Штовба С. Д., Краснов А. В. // Exponenta Pro. − 2003. − №4. − С. 70–75.

16 Ушаков С. А. Разработка и исследование алгоритмов решения задач распознавания на основе искусственных иммунных систем : дис. канд. техн. наук : 05.13.17 / Ушаков Станислав Андреевич. – Воронеж, 2016. – 138 с.

17 Кушнир Н. В. Искусственные иммунные системы: обзор и современное состояние / Кушнир Н. В., Кушнир А. В., Анацкая Е.В., Катышева П. А., Устинов К. Г. // Научные труды КубГТУ. − 2015. − №12. − С. 1–10.